Performance Evaluation of DTSN in Wireless Sensor Networks

Francisco Rocha, António Grilo, and Paulo Rogério Pereira

Abstract— **The guaranteed delivery of critical data is an essential requirement in most Wireless Sensor Network (WSN) applications. The paucity of energy, communication, processing and storage resources in each WSN node causes the TCP transport model (widely used in broadband networks) to be inefficient in WSNs, a reason why new WSN-specific reliable transport protocols have been proposed in the past few years. This paper presents one of these protocols, the Distributed Transport for Sensor Networks (DTSN). DTSN is able to efficiently support unicast communications in WSNs due to its capabilities to tightly control the amount of signaling and retransmission overhead. The basic loss recovery algorithm is based on Selective Repeat ARQ, employing both positive and negative acknowledgements. Caching at intermediate nodes is used to avoid the inefficiency typical of the strictly end-to-end transport reliability commonly assumed in broadband networks. DTSN is currently implemented in TinyOS. Preliminary simulation results using this code show that DTSN is quite efficient providing block oriented reliability, while the caching mechanism employed in DTSN decreases packet delay for large hop distances.**

Index Terms — **Energy-efficiency, Reliable Data Transport, Quality of Service, Wireless Sensor Networks**

I. INTRODUCTION

THE guaranteed delivery of critical data is an essential requirement in most Wireless Sensor Network (WSN) applications. Illustrative examples are: battlefield surveillance, intrusion detection and E-health monitoring applications, where critical alerts must be timely and reliably delivered to the monitoring stations that act on those data; and industrial control applications, where commands must be timely and reliably delivered to the actuators (e.g., robotic arm). Moreover, while in the beginning WSNs were though to convey very simple data such as alerts, actuator commands and physical measurements, the recent availability of low-cost miniaturized hardware such as CMOS cameras and microphones have led to the possibility of transmission of larger data blocks filled with sound samples and/or images. Given the paucity of resources and high bit error rate (BER) values typically featured by these networks, bulky data delivery can only be effected if the bandwidth utilization is maximized. This entails minimizing the end-to-end packet loss ratio (PLR), which is a traditional responsibility of the transport function.

Proven transport protocols like TCP [1], designed to support user applications in infrastructure networks, usually present significant inefficiencies when employed without considerable modification in WSN systems. Several proposals have been made for alternative reliable transport protocols, but only a few of them are suited to support block-oriented data delivery. The shortcomings found in these existent proposals led to the development of the Distributed Transport for Sensor Networks (DTSN) protocol [2], which is based on the following principles:

- Reliable transmission of block-oriented data;
- Energy-efficiency: to avoid useless wasting of energy resources through minimization of the control and retransmission overhead;
- Distributed Functionality: to exploit the WSN storage resources in a cooperative way, so that the scalability of WSN operation in a multihop environment is increased.

This paper presents a performance evaluation of DTSN based on its TinyOS [3] implementation developed in FP6 project UbiSec&Sens.

The rest of the paper is structured as follows. Section 2 presents the most relevant related work; section 3 describes the DTSN protocol; section 4 presents the simulation results; section 5 concludes the paper and lists some topics of ongoing and future work.

II. RELATED WORK

There are several proposals of reliable transport protocols for WSNs. This section only covers the proposals for block-oriented reliable transport that were considered more relevant for the development of DTSN.

Reliable Multi-Segment Transport (RMST) [4] was designed to work on top of the Directed Diffusion [5]. It offers two simple services: data segmentation/reassembly and guaranteed delivery. RMST can operate end-to-end or in a store-and-forward mode where intermediate nodes care to receive all the fragments of a block before forwarding it. RMST is affected by various weaknesses that make it somewhat unreliable and inefficient. First of all, there is no full guarantee of delivery. Since it only uses negative acknowledgements (NACKs) for loss recovery, when none of the fragments of a data block are received by an intermediate or destination node, the loss can not be detected

---

Manuscript received November 18, 2007. This work was supported in part by the EuroFGI (“Design and Engineering of the Future Generation Internet”). Apart from this, the European Commission has no responsibility for the content of this paper.

F. Rocha, A. Grilo and P. Pereira are with INESC-ID, Rua Alves Redol, nº 9, Lisboa, Portugal (phone: +351-213100226; fax: +351-213145843; e-mails: rocha.francisco@gmail.com, antonio.grilo@inesc-id.pt, prbp@inesc.pt).

because the sink is not aware of the transmission and there is no end-to-end positive ACK to finalize the transaction.

Distributed TCP Caching (DTC) [6] is a TCP enhancement to make it more efficient in WSNs. It improves the transmission efficiency by compressing the headers and the use of caching at selected nodes in the path from source to destination, thus minimizing end-to-end retransmissions. DTC is fully compatible with TCP, leaving the endpoints of communication unchanged – it only requires changes in the logic of intermediate nodes. The use of caching significantly improves the efficiency of packet loss recovery, minimizing the energy spent with retransmissions. However, like in TCP, DTC features an inefficiency associated with the transmission of unnecessary positive ACKs, which is totally controlled by the receiver.

Pump Slow Fetch Quickly (PSFQ) [7] is a protocol primarily designed to offer downstream multicast guaranteed delivery for dynamic code update, though it can also be used for upstream unicast communication. It supports the reliable dissemination of consistent data blocks along the network within a user defined scope. Its behavior fluctuates between multihop forwarding during the data dissemination phase (Pump phase), and store and forward when losses are detected. Upon loss detection, forwarding is stopped and recovery phase (Fetch phase) is started. Data is disseminated through broadcasting during a pre-defined time interval at the end of which flow control is applied. That approach allows to stop the propagation of gaps in the fragment sequence as soon as possible and to quickly recover missing fragments from neighbors. Every node, in fact, stores the received data to be able to provide missing fragments to requesting neighbors. The use of flow control at every hop introduces a high delay in the communication. Besides, the fact that PSFQ only employs NACK feedback (like RMST) causes it to be unable to detect the loss of all fragments of a block at once. Regarding intermediate caching, data is reconstructed at each hop. While this makes sense for dynamic code update (i.e., each node must get all the executable code fragments), that can be very limiting for other applications (audio or image transmission) since it poses significant requirements on node storage capabilities.

III. DISTRIBUTED TRANSPORT FOR SENSOR NETWORKS

The DTSN specification [2] can be divided into a full reliability service, and a differentiated reliability service. The latter is based on the former and is able to support several reliability grades based on the integration of partial buffering at the source, intermediate node caching and erasure coding. This paper focus on the full reliability service.

This service was thought for critical data transfer requiring end-to-end full reliability. The latter is achieved by a Selective Repeat Automatic Repeat reQuest (ARQ) mechanism that uses both negative acknowledgement (NACK) and positive acknowledgement (ACK) packets. Soft requirements of routing path stability and bi-directionality are placed on the routing layer in order to leverage the intermediate node optional cache mechanism of DTSN for performance improvement, although DTSN is able to operate end-to-end as well.

In DTSN, a session is a source/destination relationship univocally identified by the tuple <source address, destination address, application identifier, session number>, designated the session identifier. The session is soft-state by nature both at the source and the destination, being created when the first packet is processed and terminated upon the expiration of an activity timer (provided that no activity is detected and there are no pending delivery confirmations. A randomly chosen session number is appended in order to unambiguously distinguish between successive sessions sharing the endpoint addresses and application identifier. Within a session, packets are sequentially numbered. The Acknowledgement Window (AW) is defined as the number of packets that the source transmits before generating a confirmation request (Explicit Acknowledgement Request – EAR). The output buffer at the sender works as a sliding window, which can span more than one AW. The size of the output buffer and of the AW depend on the specific scenario, namely on the memory constraints of individual nodes.

The DTSN session management algorithm at the source works as follows. The source transmits each packet coming from the higher layers, storing it in the output buffer, so that it can be retransmitted latter if required. Upon the transmission of each set of packets equal to the AW size, or when the output buffer is full, or when the higher layer protocols have not sent any data during a predefined timeout period, the source requests a delivery confirmation message from the destination by means of an EAR. This may take the form of a bit flag piggybacked in the last data packet (e.g., confirmation request due to the AW size being reached or the output buffer becoming full) or an independent packet (e.g., confirmation request due to the expiration of the EAR timer). Each time a confirmation message (either ACK or NACK) is received, the source frees the output buffer entries whose delivery is confirmed. The reception of an ACK means that there are no gaps in the sequence of packets sent before the respective EAR. On the other hand, a NACK includes a bitmap, where each bit represents a different sequence number (starting from a base sequence number indicated in the packet header) and indicates whether the corresponding packet was correctly received or not. Its reception causes the source to retransmit the data packets that were not delivered successfully. An EAR is sent after retransmission, which may be piggybacked in the last of the retransmitted packets. After sending an EAR, the source launches an EAR timer. If the EAR timer expires before an ACK/NACK is received, the source retransmits the EAR packet.

The DTSN algorithm at the destination works as follows. Upon reception of a data packet with a new session identifier, a new session record is created. If, on the other hand, the session identifier exists, but the session number is different from the recorded one, the session record is reset and the new session number replaces the old one. The destination then collects the data packets that belong to that flow, delivering in-sequence packets to the higher layer protocol. Upon reception of an EAR, the destination sends an ACK or NACK depending on the existence of gaps in the received data packet stream. Upon the expiration of an activity timer, the session record is deleted and the higher layer protocol is notified in case there are unconfirmed packets.

As already explained, the strict end-to-end transport reliability model used in TCP is not suited for WSNs because it leads to extra consumption of the scarce bandwidth and energy resources. This is caused by the fact that missing packets (and some control packets like NACKs) are retransmitted end-to-end, expending bandwidth and energy in all links/nodes in the path between source and destination. Caching at intermediate nodes is the mechanism employed by DTSN to counter this inefficiency. In DTSN, each node keeps a cache of intercepted packets, managed according to a suitable replacement policy (currently it is FIFO). The packets are stored in cache with probability *p*, and may belong to different sessions whose end-to-end routing path includes the node in question. Each time an intermediate node receives a NACK packet, it analyses its body and searches for corresponding data packets that are missing at the destination. In case a missing packet is detected, the intermediate node retransmits it. It also changes the NACK contents before resending it, changing the bitmap so that its retransmitted packets are not included (eventually, the NACK may become an ACK in case all missing packets were found in cache). In this way, the source will only have to retransmit those data packets that were not cached at intermediate nodes, decreasing the average hop length of the paths followed by retransmitted packets. Intermediate nodes also process the session number header field in data packets, replacing any cache entries whose session number is outdated.

The caching probability *p* at the intermediate node should be defined taking into account various factors like cache size, traffic load and EAR frequency. The value of *p* must be chosen to maximize the probability of cache hit for the NACK requested data along the path.

IV. SIMULATION RESULTS

Simulations of the TinyOS 2.x implementation of DTSN were conducted using the TOSSIM simulation environment.

The radiofrequency (RF) parameters of sensor nodes correspond to those of the MICAz motes developed by Crossbow [8], which support a bitrate of 250 kbps based on the IEEE 802.15.4 standard for the MAC and physical layers [9]. The simulations were made considering a linear topology of evenly spaced sensor nodes, where the transmission power is 0 dBm (maximum power for MICAz nodes), the path loss between each pair of nodes is 70 dB (this corresponds to approximately 10 meters of distance assuming a log-distance path loss model with a distance exponent of 3) and the background noise level is -110 dBm.

DSDV [10] is used as the routing protocol underlying DTSN. A continuous traffic pattern was generated at the first node in the line, with the last node as the destination. The data payload size is 2 bytes, adding to the 6 bytes of DTSN overhead, 3 bytes of DSDV overhead and 13 bytes of MAC overhead. The 6 bytes of DTSN overhead include 1 byte for flags (e.g., ACK, NACK), 1 byte for the packet sequence number and 4 bytes for the session identifier. The DTSN overhead may be reduced by fetching information of source and destination addresses from the routing layer. As this optimization depends on the specific routing protocol used, it was not used in the experiments. The DTSN transmission window size was configured as 50 packets, with two AWs of 25 packets each. The EAR timeout was set as 250 ms. The caching probability *p* (when used) is set to 1 and the cache size is enough for 50 packets.

The results allow a comparison between raw DSDV transmission (without reliable transport on top) and DTSN with and without the intermediate node caching mechanism. Each point in the graphics corresponds to an average over ten independent runs, each consisting of the transmission of 1000 packets back-to-back (subject to local interlayer flow control). The achieved packet loss ratio, throughput, average delay and per-packet overhead (measured as the total number of radio transmissions per successfully delivered packet received at the destination) are depicted respectively in Fig. 1, Fig. 2, Fig. 3 and Fig. 4 as a function of the hop distance between source and destination.

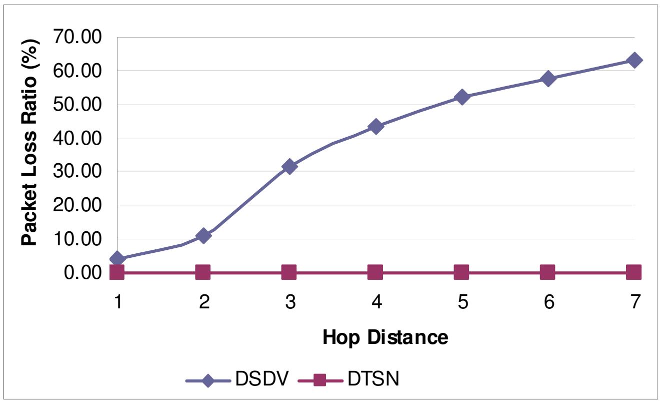

Fig. 1: Packet Loss Ratio.

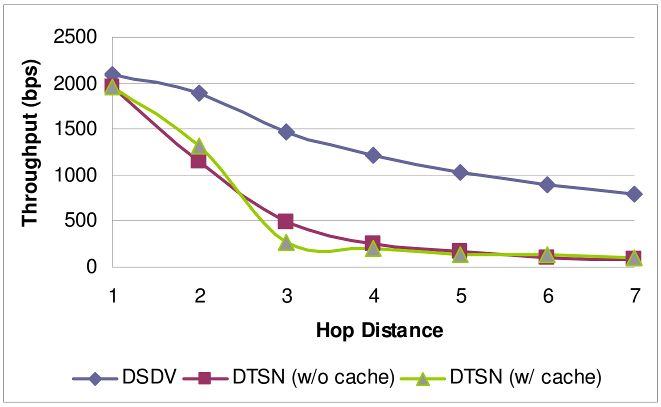

Fig. 2: Throughput.

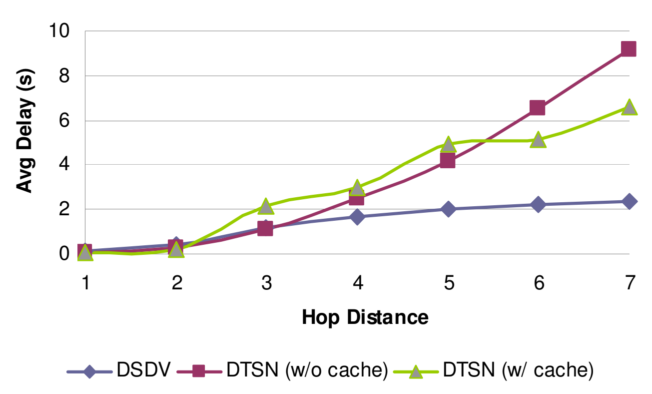

Fig. 3: Average Delay.

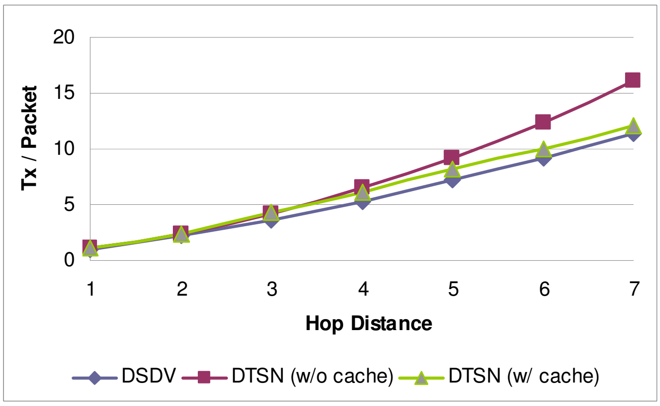

Fig. 4: Average number of RF transmissions per successfully delivered data packet.

The packet loss for DSDV increases quite steadily with the hop distance, justifying the use of a reliable transport protocol like DTSN. However, DSDV still achieves higher throughput and lower delay (the latter, for hop distances greater than 3) compared with DTSN, which mimics the difference between TCP and UDP in IP networks. Regarding the overhead, although DSDV clearly presents a lower number of transmissions per packet compared with DTSN without cache, the difference is not very significant when compared with DTSN with the cache mechanism turned on. For a hop distance greater than 4, the cache mechanism starts to pay-off in terms of overhead. In terms of average delay, it only starts to pay-off for a hop distance greater than 5. This surprising result is due to the interference between NACK and data packets retransmitted at intermediate nodes when the cache mechanism is turned on. NACK and data packets are simultaneously forwarded in different directions, interfering with each other and causing a higher NACK loss probability. In fact, the DTSN configuration without cache presented a significantly lower number of EAR timeouts, and the analysis of the simulation traces confirmed this explanation. This problem is currently being investigated so that the caching mechanism is enhanced to avoid this interference.

V. CONCLUSIONS

This paper has presented a performance evaluation of the Distributed Transport for Sensor Networks (DTSN). This reliable transport protocol for wireless sensor networks minimizes control overhead by placing the control of the session at the sender, which explicitly requests the data delivery confirmations from the receiver. A caching mechanism at the intermediate nodes attempts to minimize end-to-end retransmissions.

The performance evaluation of DTSN was based on a TinyOS 2.x implementation. The simulation results attest the effectiveness of DTSN to minimize the number of RF transmissions per packet. Although the caching mechanism proved to be effective for large hop distances, interference between control and data packets leads to an higher number of NACK packets being lost, leveling the performance of the DTSN configurations with and without cache. This problem is currently being tackled. After the caching mechanism is enhanced, future work will focus on the evaluation of DTSN for different packet sizes, network topologies and traffic load values, comparison with other WSN transport protocols, test with other routing and MAC protocols, as well as the implementation and evaluation of the differentiated reliability service in Wireless Multimedia Sensor Networks.

REFERENCES

[1] J., Postel, “Transmission Control Protocol”, RFC 793, IETF, September 1981.

[2] B. Marchi, A. Grilo, and M. Nunes, “DTSN – Distributed Transport for Sensor Networks”, Proceedings of the IEEE Symposium on Computers and Communications (ISCC’07), Aveiro, Portugal, 2007.

[3] J. Hill, R. Szewczyk, A.Woo, S. Hollar, D. Culler, K. Pister, “System Architecture Directions for Networked Sensors”, Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), November 2000.

[4] F. Stann, and J. Heidemann, “RMST: Reliable Data Transport in Sensor Networks”, Proceeding of the *1st IEEE International Workshop on Sensor Net Protocols and Applications*, IEEE, Anchorage, Alaska, USA, May 2003, pp. 1-11.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks”, *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCON)*, August 2000.

[6] A. Dunkels, J. Alonso, T. Voigt, and H. Ritter, “Distributed TCP Caching for Wireless Sensor Networks”, *Proceedings of the 3rd Annual Mediterranean Ad-Hoc Networks Workshop*, Bodrum, Turkey, June 2004.

[7] C.-Y. Wan, A. Campbell, and L. Krishnamurthy, “PSFQ: A Reliable Transport Protocol for Wireless Sensor Networks”, *Proceedings of WSNA’2002*, Atlanta, Georgia, USA, 2002.

[8] "MICAz datasheet", Crossbow, 24/4/2007, http://www.xbow.com/Products/productdetails.aspx?sid=164 last accessed on 3rd January 2008.

[9] IEEE Std. 802.15.4, “Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)”, 2003.

[10] C. Perkins, and P. Bhagwat, “Highly Dynamic Destination-sequenced Distance-vector routing (DSDV) for Mobile Computers”, in SIGCOMM, pp. 234–244, 1994.